server name as associated with a separate application independent of the physical host computer.

REMARKS

Claims 1-23 are pending in the present application. Claims 1, 7, 9, 12, and 18 were amended. No claims were added or canceled. Reconsideration of the claims is requested.

I. 35 USC § 102 ANTICIPATION

The Examiner has rejected claims 1-23 under 35 USC § 102(e) as being anticipated by *Kandasamy* (U.S. Pat. No. 6,219,799 B1). This rejection is respectfully traversed.

Claim 1, as amended, reads:

> 1. A method for configuring a server in a distributed data processing system, the method comprising the computer-implemented steps of:
> registering a primary server name for the server at the application protocol layer;
> registering a secondary server name for the server at the application protocol layer; and
> responding to requests directed to either the primary server name or the secondary server name, wherein clients communicating with the server perceive each server name as associated with a separate data structure independent of the physical host computer.

In rejecting claim 1, the Examiner states:

> As claim 1 [sic], Kandasamy teaches a method for configuring a server in a distributed data processing system [abstract, lines 7-11], the method comprising the computer implemented steps of:
> registering a primary server name for the server; registering a secondary server name for the server; and responding to the requests directed to either the primary server name or the secondary server name [col., line 36-30; col. 2, lines 45-54; col. 3, lines 16-20; 31-32].

(Office Action dated 9/25/01, page 2.)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

Kandasamy appears similar to the present invention because both teach using multiple network names with servers. However, an important difference between *Kandasamy* and the present invention is the protocol layer at which the server names are registered. *Kandasamy* registers multiple server names at the import driver level. By contrast, the present invention registers server names at the application level. The consequences of this difference are very important and even take the present invention in the opposite direction of *Kandasamy*.

Because Kandasamy couples the multiple server names with the protocol driver, it is merely extending the functions of the driver. The uses of *Kandasamy*'s invention are pretty much limited to the fail-over functions described in the patent. Coupling multiple server names with the import driver creates a union of shared resources between two servers during a fail-over situation, and makes the two servers appear like a single server to outside clients because the separate names are associated with the same driver. This has the effect of collapsing the two servers into one.

By contrast, the present invention couples the server names with the applications. This approach creates separate data file structures that are distinguishable from each other, even in a fail-over situation. Because each server name in the present invention is coupled to a specific application, information is correctly exported to clients depending on the name requested by the clients. In addition, because the names are coupled to the application, rather than the driver, the clients only perceive the specific data structure associated with the name that they are requesting information from, without any knowledge of the physical host machine on which the data structure is located.

Since the server names are coupled at the application level in the present invention, the names and their associated data structures can be added to and removed from host machines without clients being aware of such movements, because the clients only perceive the name and application, not the specific machine. This allows the present invention to be used for load balancing between computers. For example, if a computer is running four different applications and the processing load for that computer becomes too great, a system administrator can remove a name and its associated application to another computer without the clients in the network being aware of the physical transfer. *Kandasamy* cannot be used for such purposes and can only be used for fail-over situation, because *Kandasamy* teaches coupling the names at the driver level.

Moreover, the method described in *Kandasamy* requires the servers to be identical because the resources of the servers will have to be combined during a fail-over. The present invention does not have this requirement.

Kandasamy not teach all of the limitations of claim 1, nor does Kandasamy suggest them. In addition, the teachings of *Kandasamy* cannot be extended to create the features of the present invention by simply applying Kandasamy at the application protocol layer. This is because *Kandasamy* specifically teaches and claims a system using the Microsoft Windows NT operating system. In order to modify the teachings of *Kandasamy* to implement the present invention, a person would have to change the file service code of Windows NT. This would require changes in the Windows NT source code, which can only be made by Microsoft Corporation, which owns the code.

In rejecting claims 2 and 3, the Examiner states:

> As claims 2 and 3 [sic], Kandasamy further teaches of comprising registering the primary server name or the secondary server name while configuring or initializing the server and reading the primary server name of the secondary server name from a configuration file or initialization file [abstract, line 8-11; col. 3, lines 16-20].

(Office Action, pp. 2-3.) This rejection is respectfully traversed for the reasons explained above.

Claims 2 and 3 read:

2. The method of claim 1 further comprising registering the primary server name or the secondary server name while configuring or initializing the server.

3. The method of claim 1 further comprising reading the primary server name or the secondary server name from a configuration file or initialization file.

As explained above, the server names are being configured and initialized at the application protocol layer, not the driver layer, as taught in *Kandasamy*.

In rejecting claim 4, the Examiner states:

> As claim 4 [sic], Kandasamy teaches that the distributed data processing system comprises a network, and wherein the requests are received from the network [fig. 1; abstract, lines 8-11; col. 3, lines 12-13].

(Office Action, p. 3.)

Claim 4 reads:

4. The method of claim 1 wherein the distributed data processing system comprises a network, and wherein the requests are received from the network.

While both inventions are used within computer networks, as explained above, registering server names at the application level has the effect of making clients within the network perceive only the name/data structure without specific knowledge of the physical host machine. Therefore, the effects on network communications are different between the two inventions.

In rejecting claim 5, the Examiner states:

> As claim5 [sic], Kandasamy further teaches of comprising the primary server name or the secondary server name with a call to a NetBIOS application programming interface [col. 2, lines 41-54, 63-67].

(Office Action, p. 3.)

Claim 5 reads:

5. The method of claim 1 further comprising registering the primary server name or the secondary server name with a call to a NetBIOS application programming interface.

The sections cited by the Examiner do not describe the limitations of claim 5. Specifically, that section of *Kandasamy* reads:

> Intelligent Network Storage Controller (INSC) will be understood as a highly available Network Attached Storage whose design will be understood to support two Computers running Microsoft Windows NT operating system. When one of the computer goes down for some reason, the other computer will take over all the functions of the failed computer. The NETBIOS and SMB services of the Microsoft Windows NT Operating system depend on the "name" of the computer to function. So, to provide this functionality, it is required that the computer that "takes over" should support more than one name. This calls for supporting multiple names in a single computer, as a feature hereof.
>
> To support Multiple names in Windows NT Operating system environment, and provide the SMB file services, the following functionalities are required:
>
> A. Register the Computer name using the NETBIOS Protocol specified in the RFC 1001, and RFC 1002, and respond to name queries from the other computers in the network;
>
> B. The SMB server should be listening for the network requests on that computer name. Requirement A is effected using a driver module that registers the second name with the appropriate NETBIOS Transport. To satisfy requirement #B. (Col. 2, lines 42-67)

Kandasamy merely mentions NETBIOS as the communication protocol used for network, but it does not teach a call to an application programming interface (API). This makes sense when one remembers that *Kandasamy* is only being applied at the driver level, not the application level.

In rejecting claim 6, the Examiner states:

> As claim 6 [sic], Kandasamy teaches that the server comprises a plurality of secondary server names [col. 3, lines 6-7].

(Office Action, p. 3.)

Claim 6 reads:

> 6. The method of claim 1 wherein the server comprises a plurality of secondary server names.

While both inventions teach the use of multiple server names, as explained above, *Kandasamy* teaches coupling these names with the protocol driver, whereas the present invention teaches coupling the names with the applications.

In rejecting claims 7 and 8, the Examiner states:

> As claims 7 and 8, Kandasamy teaches a method for reconfiguring servers in a distributed data processing system [abstract, lines 7-11], the method comprising the computer-implemented steps of:
>
> registering a first primary server name for a first server; registering a second primary server name for a second server [col. 3, lines 8-11]; determining that the first server requires reconfiguration in response to a determination that the second server requires fail-over support by the first server [col. 3, lines 14-15]; registering for the first server a secondary server name that is identical to the second primary server name [col. 3, lines 17-20; and responding by the first server to requests directed to either the first primary server name or the secondary server name [col. 2, lines 45-54, col. 3, lines 31-32].

(Office Action, p. 3.) This rejection is respectfully traversed for the reasons stated above.

Claims 7 and 8, as amended, read:

> 7. A method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:
>
> registering a first primary server name for a first server at the application protocol layer;
>
> registering a second primary server name for a second server at the application protocol layer;
>
> determining that the first server requires reconfiguration;
>
> registering for the first server a secondary server name that is identical to the second primary server name; and
>
> responding by the first server to requests directed to either the first primary server name or the secondary server name, wherein clients communicating with the server perceive each server name as associated with a separate data structure independent of the physical host computer, and wherein the clients perceive responses from the second primary name on the second server and responses from the identical secondary name on the first server as coming from the same data structure.
>
> 8. The method of claim 7 wherein the first server is reconfigured in response to a determination that the second server requires fail-over support by the first server.

While both inventions do provide fail-over functions, the present invention registers the multiple server names at the application protocol layer, whereas *Kandasamy* works at the driver level. The effects of this difference have already been explained above in relation to claim 1. In a fail-over situation, *Kandasamy* creates a union of shared resources, which effectively collapses two servers into one. The present invention maintains separate data/resource structures, even during fail-over.

In rejecting claims 9-11, the Examiner states:

> As claims 9-11, Kandasamy teaches a method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:
> registering a first primary server name for a first server; registering a secondary server name for the first server; responding to requests directed to either the first primary server name or the secondary server name [col. 1, lines 36-39; col. 2 lines 45-54; col. 3, lines 14-15]; determining that the first server requires reconfiguration [col. 3, lines 14-15]; deregistering the secondary server name for the first server [col. 3, lines 24-25];
> registering for a second server a second primary server name that is identical to the secondary server name [col. 3, lines 3-5].

(Office Action, p. 4.) This rejection is respectfully traversed for the reasons stated above.

Claims 9-11, as amended, read:

> 9. A method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:
> registering a first primary server name for a first server at the application protocol layer;
> registering a secondary server name for the first server at the application protocol layer;
> responding to requests directed to either the first primary server name or the secondary server name;
> determining that the first server requires reconfiguration;
> deregistering the secondary server name for the first server; and
> registering for a second server a second primary server name that is identical to the secondary server name;
> wherein clients communicating with the server perceive each server name as associated with a separate data structure independent of the physical host computer, and wherein the clients perceive responses from the second primary name on the second server and responses from the secondary name on the first server as coming from the same data structure.

10. The method of claim 9 wherein the secondary server name is deregistered prior to connecting the second server to a network in the distributed data processing system.

11. The method of claim 9 wherein the second primary server name is registered prior to connecting the second server to a network in the distributed data processing system.

The sections of *Kandasamy* cited by the examiner refer to a fail-over procedure in which the resources of two servers are united on one server. Claims 9-11 describes an offloading process, which goes in the opposite direction of a fail-over. The methods of claims 9-11 allow for the transference of data structures from one computer to another without the client machines being aware of the change in physical host computers. Not only does *Kandasamy* not teach these features, it cannot be used for this purpose because it does not teach coupling the server names with the applications, as is done in the present invention.

In regard to the remaining claims, the Examiner states:

> As claims 1-17, Kandasamy teaches the claimed method of steps. Therefore, Kandasamy teaches the claimed system for carrying out the method of steps.
>
> As claims 18-23, Kandasamy teaches the claimed method of steps. Therefore, Kandasamy teaches the claimed computer program product for carrying out the method of steps.

(Office Action, p. 4.) This rejection is respectfully traversed for the reasons explained above in regard to the method claims.

Therefore, the rejection of claims 1-23 under 35 USC § 102(e) has been overcome.

II. CONCLUSION

It is respectfully urged that the subject application is patentable over *Kandasamy* and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: Jan 25, 2002

Respectfully submitted,

Christopher P. O'Hagan

Christopher P. O'Hagan
Reg. No. 46,966
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
ATTORNEY FOR APPLICANT

APPENDIX OF CLAIMS

The following are marked up versions of the amended claims:

1. A method for configuring a server in a distributed data processing system, the method comprising the computer-implemented steps of:

registering a primary server name for the server at the application protocol layer;

registering a secondary server name for the server at the application protocol layer; and

responding to requests directed to either the primary server name or the secondary server name, wherein clients communicating with the server detect each server name as associated with a separate application independent of the physical host computer.

7. A method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:

registering a first primary server name for a first server at the application protocol layer;

registering a second primary server name for a second server at the application protocol layer;

determining that the first server requires reconfiguration;

registering for the first server a secondary server name that is identical to the second primary server name; and

responding by the first server to requests directed to either the first primary server name or the secondary server name, wherein clients communicating with the server detect each server name as associated with a separate application independent of the physical host computer, and wherein the clients detect responses from the second primary name on the second server and responses from the identical secondary name on the first server as coming from the same application.

9. A method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:

registering a first primary server name for a first server at the application protocol layer;

registering a secondary server name for the first server at the application protocol layer;

responding to requests directed to either the first primary server name or the secondary server name;

determining that the first server requires reconfiguration;

deregistering the secondary server name for the first server; and

registering for a second server a second primary server name that is identical to the secondary server name;

wherein clients communicating with the server detect each server name as associated with a separate application independent of the physical host computer, and wherein the clients detect responses from the second primary name on the second server and responses from the identical secondary name on the first server as coming from the same application.

12. A data processing system comprising:

registering means for registering a primary server name for a server and for registering a secondary server name for the server at the application protocol layer; and

responding means for responding to requests directed to either the primary server name or the secondary server name, wherein clients communicating with the server detect each server name as associated with a separate application independent of the physical host computer.

18. A computer program product on a computer readable medium for use in a data processing system, the computer program product comprising:

instructions for registering a primary server name for a server and a secondary server name for the server at the application protocol layer; and

instructions for responding to requests directed to either the primary server name or the secondary server name, wherein clients communicating with the server detect each server name as associated with a separate application independent of the physical host computer.